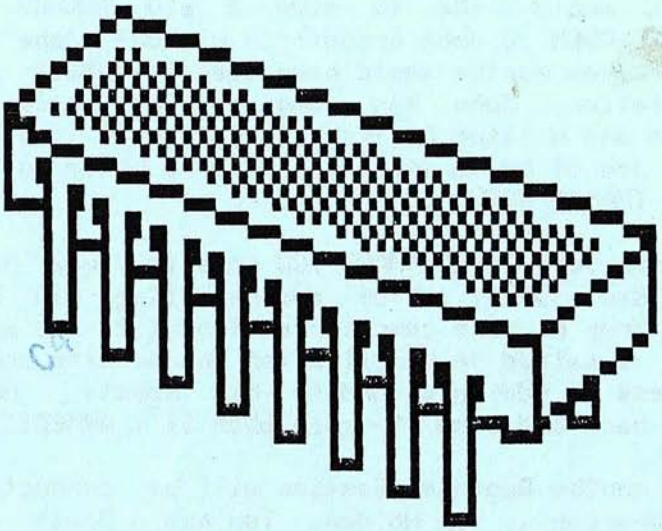
ATARI COMPUTER CLUB OF OKC INC.

P. O. BOX 32672
OKLAHOMA CITY, OK. 73128
NEWSLETTER VOL. #VII ISSUE #10

DO YOU SEE



WHAT IC ?

MEETING AT OKLA. MILITARY DEPT.

3501 MILITARY CIRCLE (36TH & GRAND)
MEETING TIME: 6:15 TO 9:00 PM
MEETING DATE: MAY 12TH, 1987

PRESIDENTIAL NOTES

by Bob Bewley

I think SPRING has finally SPRUNG upon me!!! I have found myself working in the great outdoors and NOT using my computer very much. In fact, I've only used my computer about 4 or 5 times last month. I know I said I wouldn't let that happen to me... but when I can't see my children playing in the yard, I know it's time to cut the grass!!! Anyway, now that I have the major work out of the way, I'll probably get back to the computer more often this month.

I would like to give a BIG THANKS and HARDY PAT-ON-THE-BACK to John Brandt!!! Without John's articles these past few months would have been the WORST months for our newsletter. John has come to the rescue with his knowledge and writing habits. It's members like John that make the job of being (Acting) Editor a Whole Lot easier!!! Again... THANKS JOHN!!!

Another very large THANK YOU goes to Virgil Holden. I think he did a GREAT job of demonstrating his ROBOT. I think he brought more computer equipment to the meeting than I did!!! I talked to Virgil after the meeting and he is in the process of adding an ARM to his Robot. Maybe he'll bring it back and show it again when it's ARMED!!! (ha-ha)

This months Beginner Session will be conducted by our Super Librarian... Tom Holden. Tom has a Great version of the Wheel of Fortune Game on this month's Disk of the Month (the only thing missing is Vana White!!!). The game comes with approximately 1300 puzzles with the capability to add as many as you want. During the Beginner's Session, Tom is going to show you how to ADD YOUR OWN PUZZLES to the Wheel of Fortune.

The Regular Session will consist of software demonstrations from LUCASFILM. These include games like Eidolon, Ballblazer and Rescue on Fractalus. These games have VERY impressive graphics and sound, YOU HAVE TO SEE THEM TO BELIEVE THEM!!!

Don't forget about the door prizes again this month... We have a few more STAR RAIDERS and JINGLE DISKS to give away. Someone missed out on winning last month because they left early. DON'T MISS OUT THIS MONTH!!!

See you at the meeting

TREASURER'S REPORT

by Becky Pound

Beginning Balance			$307.17
Debits		$116.97	
Printing	$30.00		
Stamps	$22.00		
Printer Ribbons	$12.84		
ANTIC Disks	$39.63		
Door Prizes	$12.50		
Credits		$49.00	
D.O.M.	$32.00		
Blank Disks	$12.00		
Dues	$ 5.00		
Ending Balance			$239.20

RECORDINGS FROM THE LIBRARIAN

by Tom Holden

I added eight new 8-bit disks (and 1 ST disk), to the Club Library, ANTIC V2-05 thru V2-08 (Aug. to Oct. 1983), V5-11 (March 1987), V6-01 (May 1987) and A.N.A.L.O.G. Number 52 and 53 (March & April 1987).

The SYSOP on BALLPARK gave us ANTIC March 1987, and the club bought May-87 ANTIC from MERITS (to get the "WORDS ARE FUN" game). The 1983 ANTIC's came from ANTIC, as they are having a SALE where you can get six months of old ANTIC disks for $20.00. The new A.N.A.L.O.G. disks came from our club's subscription, however the last issue I got was a 3 1/2 inch ST-Disk, not our normal 5 1/4 inch (8-Bit) disk. So if any of our members have an ST, our library now has one A.N.A.L.O.G. ST disk (Feb-87).

The WORDS ARE FUN program will be a boot disk on the back side of MAY'S Disk-of-the-Month. I have modified this game so that there are more random sayings and comments. Also added about 125 more puzzles to the data files, for about 1,400 total puzzles.

Remember, if you want to make a copy of any of the programs in the Club Library, you can take your blank di and go to MERIT'S (50th & N. Portland) and make a copy, o give me a blank disk and a list of the files you want at any meeting, or come over to my house.

All copies are Free if you are a club Member and you bring the disk, else the club charges $1.00 (for the blank disk). The D-O-M cost's $4.00 or you can buy a D-O-M Card for $20.00, which allows you to get SIX D-O-M disks for the price of FIVE.

WOW wasn't that ROBOT Demo something! Thanks go out to Virgil Holden for showing us how to make his ATARI Computer communicate with his Robot. Have you ever seen a robot do so many things?

The only missing ANTIC disks are Feb. & March 1983 and A.N.A.L.O.G. # 44 thru 49 (July to Dec 1986).

See you at the next meeting, Tom

* ACCOKC DISK LIBRARY *

DISK#	PROGRAM	EXT	SIZ	PROGRAM TYPE
A 52	BOOTCAMP	M65	073	BINARY FILE
A 52	DEVILS	BAS	088	BASIC BOOT
A 52	DUMPMATE	AMA	025	UTILITY
A 52	DUMPMATE	BAS	045	BASIC BOOT
A 52	MIDAS	OBJ	109	BINARY BOOT
A 52	MIDAS1	ACT	098	ACTION FILE
A 52	MIDAS2	ACT	086	ACTION FILE
A 52	RAMBUG	M65	182	BINARY FILE
A 52	RAMBUGII	OBJ	036	BINARY BOOT
A 52	SHAPES	STB	023	
A 52	VBI1	BAS	026	BASIC BOOT
A 52	VBI2	BAS	026	BASIC BOOT
A 52	VBI3	BAS	027	BASIC BOOT
D 21	DEVILS	BAS	088	BASIC BOOT
D 21	DUMPMATE	BAS	045	UTILITY
D 21	MIDAS	OBJ	109	BINARY BOOT
D 21	PICTURE		062	DEMO PROGRAM
D 21	RAMBUGII	OBJ	036	BINARY BOOT
D 21	SLITHER	OBJ	035	BINARY BOOT
D 21	STARLANE	BAS	087	BASIC BOOT
D 21	TABLETYP	BAS	056	UTILITY
D 21	WARRIORK	ING	DSK	BINARY BOOT
D 22	WORDSARE	FUN	DSK	BASIC BOOT
V2-05	CAPITAL	PLT	010	PILOT FILE
V2-05	ESCHER	BAS	048	BASIC BOOT
V2-05	FUJI1	BAS	026	BASIC FILE
V2-05	FUJI2	BAS	076	BASIC FILE
V2-05	KEYSTROK	BAS	114	BASIC BOOT
V2-05	MZMANIAC	BAS	049	BASIC BOOT
V2-05	PICUTIL	BAS	081	UTILITY
V2-05	RESIST2	LST	006	SUBROUTINE
V2-05	RESISTOR	BAS	034	BASIC BOOT
V2-05	TALK	ASM	020	BINARY FILE
V2-05	TALK	BAS	023	BASIC BOOT
V2	TRAIN	PLT	025	PILOT FILE
V2	TURNCOAT	BAS	010	BASIC BOOT
V2-06	BINAUTO	ASM	046	BINARY FILE
V2-06	CARDFILC	BAS	026	BASIC FILE
V2-06	CARDFILD	BAS	028	BASIC FILE
V2-06	FORTHASM	4TH	020	
V2-06	HOOKEY	BAS	051	BASIC BOOT
V2-06	OBJ2STR	BAS	019	BASIC BOOT
V2-06	OBJ2STR	SYN	005	
V2-06	PATTERNS	BAS	065	BASIC BOOT
V2-06	PMDEMO	BAS	008	DEMO PROGRAM
V2-07	ALPHARUN	BAS	054	BASIC FILE
V2-07	ARITHMTX	BAS	037	BASIC BOOT
V2-07	AUTOCAS	BAS	018	UTILITY
V2-07	DOSSECT	4TH	015	
V2-07	LOCO	PLT	014	PILOT FILE
V2-07	MAXMIND	ASM	121	BINARY FILE
V2-07	MAXMIND	OBJ	011	BINARY BOOT
V2-07	MICROLDR	ASM	004	BINARY FILE
V2-07	MICROLDR	BAS	019	BASIC BOOT
V2-07	MICROLDR	LST	010	TEXT FILE
V2-07	MTSKIING	BAS	017	BASIC BOOT
V2-07	PRINTER	ASM	031	BINARY FILE
V2-07	PRINTER	BAS	008	BASIC BOOT

* ACCOKC DISK LIBRARY *

DISK#	PROGRAM	EXT	SIZ	PROGRAM TYPE
V6-01	WFPIZL4	000	023	TEXT FILE
V6-01	WFPIZL5	000	023	TEXT FILE
V6-01	WFPIZL6	000	023	TEXT FILE
V6-01	WFPIZL7	000	024	TEXT FILE
V6-01	WFPIZL8	000	023	TEXT FILE
V6-01	WFPIZL9	000	023	TEXT FILE
V6-01	WFUN	TXT	011	TEXT FILE
V6-01	WORDFONT		011	SUBROUTINE
V6-01	WORDFUN		166	ARCADE GAME
V2-07	RESET	LST	002	DATA FILE
V2-07	TIMEFIRE	BAS	010	BASIC BOOT
V2-07	TRAKSTAR	BAS	062	BASIC BOOT
V2-08	AIRRAID	BAS	074	BASIC BOOT
V2-08	CHORDS	BAS	023	BASIC FILE
V2-08	DRAWTEXT	ASM	025	BINARY FILE
V2-08	DRAWTEXT	BAS	035	BASIC BOOT
V2-08	RNDMUSIC	BAS	014	BASIC BOOT
V2-08	SMPSYNTH	BAS	033	BASIC BOOT
V2-08	SNDDEMO	BAS	020	DEMO PROGRAM
V2-08	SNDEDTR	ASM	053	BINARY FILE
V2-08	SNDEDTR1	BAS	010	BASIC FILE
V2-08	SNDEDTR2	BAS	006	BASIC FILE
V2-08	SNDEDTR3	BAS	048	BASIC FILE
V2-08	TAG	BAS	037	BASIC BOOT
V2-08	TOOT	LGO	030	LOGO FILE
V5-11	CONSOLE	LST	006	SUBROUTINE
V5-11	CONSOLE	M65	021	BINARY FILE
V5-11	DVORAK	BAS	020	BASIC BOOT
V5-11	DVORAK	EXE	004	SUBROUTINE
V5-11	DVORAK	M65	035	BINARY FILE
V5-11	JSTICK	LST	004	UTILITY
V5-11	JSTICK	M65	012	BINARY FILE
V5-11	MULTIAUT	BAS	013	BASIC BOOT
V5-11	NEWOWN12	BAS	020	BASIC BOOT
V5-11	PUZZLE1	DAT	004	DATA FILE
V5-11	PUZZLE2	DAT	011	DATA FILE
V5-11	SCREEN	BAS	040	UTILITY
V5-11	VECTRON	BAS	050	BASIC BOOT
V5-11	WAGES	BAS	041	BASIC BOOT
V5-11	WORD	BAS	055	BASIC BOOT
V6-01	AROGUE	BAS	125	BASIC FILE
V6-01	AROGUE	FNT	009	SUBROUTINE
V6-01	HELP		064	SUBROUTINE
V6-01	HOP	BAS	021	BASIC FILE
V6-01	NEWOWN14	BAS	036	BASIC FILE
V6-01	POKER	BAS	089	BASIC FILE
V6-01	PORT	TXT	029	TEXT FILE
V6-01	SOUNDMEN	BAS	030	BASIC FILE
V6-01	SOUNDST	BAS	013	BASIC FILE
V6-01	TOWERS	BAS	081	BASIC FILE
V6-01	WFPIZL1	028	023	TEXT FILE
V6-01	WFPIZL10	000	023	TEXT FILE
V6-01	WFPIZL11	000	023	TEXT FILE
V6-01	WFPIZL12	000	022	TEXT FILE
V6-01	WFPIZL13	000	013	TEXT FILE
V6-01	WFPIZL2	000	024	TEXT FILE
V6-01	WFPIZL3	000	023	TEXT FILE

SUPPLEMENT MAY 12 1987

BASIC History

by John Brandt

By popular demand, I have written this article about Atari BASIC. Some of our members are not familiar with the evolution of Atari BASIC; in particular, they don't know about the various revisions of BASIC and what differences exist bewteen them. So, this article will recap the various versions of Atari BASIC, discuss the known bugs in each version, and what to do about them.

Atari has marketed three versions of Atari BASIC, as well as two versions of Microsoft BASIC. This article only discusses Atari BASIC; few (if any) of our members even have Microsoft BASIC, let alone use it with any regularity.

The original Atari BASIC is known simply as "revision A." It was distributed on cartridge for the Atari 400, 800, and 1200 computers because these computers did not have built-in BASIC. It contains several bugs, but most are relatively obscure, so it's still a quite useful product.

The built-in BASIC contained in most 600XL and 800XL computers, a la Commodore's once-popular VIC-20 and "64" is "revision B." Since the bugs in revision A are well documented, the "old" Atari decided to fix them for the new computers, creating "Revision B." In the process, however, they introduced a couple of nasty new bugs. These bugs are harder to get around than those in rev. A.

Because of the severity of the bugs in revision B, the "new" Atari decided to create one more revision (C, of course). This is the BASIC found in the last few 800XL's and all the XE computers. Atari also makes it available on cartridge for those who have rev. A or B and wish to upgrade. All the bugs that were introduced in rev. B have been corrected in rev. C. Rev. C therefore contains the fewest bugs.

While all this was going on at Atari, the company which originally designed rev. A BASIC, Optimized Systems Software (OSS for short), released three "enhanced" versions of Atari BASIC: BASIC A+, BASIC XL, and BASIC XE. The first of

these, BASIC A+, is a disk-based BASIC which is "upward compatible" with Atari BASIC. Like Atari's rev. C, it corrects all the major bugs in rev. A. It also supports several new BASIC statements to simplify everything from structured programming to player-missile graphics, and it's somewhat faster than Atari BASIC because it includes a few of the "FASTCHIP" floating-point routines developed by Newell Industries. Its one major drawback is that its "SAVEd" files are not compatible with Atari BASIC "SAVEd" files. The latest revision allows you to LOAD files saved under Atari BASIC, but the only way to convert your programs from BASIC A+ back to Atari BASIC is to LIST the program to disk, boot up with Atari BASIC, and then re-ENTER the program. Earlier releases require this nonsense even going the other way.

OSS's next development was BASIC XL. Despite the name, it doesn't require an XL computer. It's essentially the next upgrade of BASIC A+, adding such features as (finally!) string arrays, full compatibility with SAVEd Atari BASIC files (assuming no new features were used), and a "FAST" command which speeds programs up considerably. However, it comes on OSS's "Supercartridge (tm)" rather than on disk. I still think this was done mainly to thwart pirates, but the cartridge design has advantages for us users as well: it allows you to use the 16K BASIC XL interpreter and still have 40K for DOS and your BASIC program. (With most 16K cartridges, such as Atari's "Microsoft BASIC," you only have 32K for DOS and your own program.) With the companion DOS XL, most of DOS itself can load "underneath" the cartridge, which gives you even more program space.

OSS later released an "add-on" for BASIC XL called the "BASIC XL ToolKit," which comes on disk. It contains an execute-only version of BASIC XL on disk, an "Extensions" module, and some sample programs. The execute-only BASIC allows people who don't have the BASIC XL cartridge to run BASIC XL programs. (They can't edit or list the programs, of course.) The extensions add several commands which support named procedures and string sorting, and sample programs are included to demonstrate these new commands.

The latest BASIC from OSS is BASIC XE. The name is even more confusing than BASIC XL, because you don't need an

XE computer, but you DO need an XL computer with 64K. BASIC XE is essentially BASIC XL, plus the ToolKit, with some of the commands that used to be on cartridge moved to disk (and vice versa), and one new feature: if you have a 130XE (or a compatible upgrade), you can use the "EXTEND" statement to place your strings and arrays in "extended" memory (a la the Commodore 128). This allows larger programs, since you can have up to 64K of strings and/or arrays without using up that valuable 40K of DOS/program space. (This precludes use of the DOS 2.5 RAMdisk, although if you have more than 128K you may still be able to set up a "beyond 130XE" RAMdisk.) All in all, I think BASIC XL and its ToolKit are a little better organized, so unless you really need to write BASIC programs of 40K or more, I recommend BASIC XL over BASIC XE.

Now assuming you have Atari BASIC (practically everyone does), how do you tell which revision you have? The easiest way to tell is simply to enter "PRINT PEEK(43234)" at BASIC's READY prompt. Location 43234 happens to contain a different value in each revision. The following table tells you which revisions contain what values.

Revision	Contains
A	162
B	96
C	234

Now that you know which revision you have, what bugs do you have? I'm glad you asked. Here is a list of all BASIC bugs that I know of.

Bug #1: The infamous keyboard lockup bug. (Revs. A and B)

This bug is the most serious bug in revs. A and B (that's why it's "infamous"). Sometimes, when editing a BASIC program, the program gets garbled, or the computer simply crashes.

Another effect of this bug is that string assignments involving multiples of 256 bytes don't work correctly. This is especially irritating because one of the advantages of Atari BASIC over most others (such as Microsoft) is supposed to be that you can use strings longer than 255 bytes. But

how can you use long strings if you don't know if they're going to work right?

Fortunately, the string half of this bug isn't too difficult to work around: if you're doing string assignments which might attempt to move a multiple of 256 characters, modify them so that they always move an odd number of characters. In other words, if you want to move an even number of characters, just move one "extra" character.

The editing bug is harder to avoid. The best advice, I believe, applies bug or no bug: Save early and often. There are a few things you can do, however: If you have several consecutive lines to delete or edit, save your program first. If you have rev. A, delete or edit the lines in descending order. For rev. B, do them in ascending order. Chances are, if you get by the first line, they'll all work OK. If you don't make it, however, reboot, reload your program, and skip the line you crashed on, proceeding in order with the rest. This is almost sure to work. Save your work again before making that change you skipped, however--it could still crash.

Bug #2. (Rev. A only.) Unary minus (e.g., -X), doesn't work correctly if the value being negated happens to be zero. You'll never notice this unless you try to print it out, because -0 still evaluates to zero in an expression. If you do happen to print -0, however, you'll see garbage. This is easy to work around, though: instead of PRINTing -A, use PRINT 0-A.

Bug #3. (Rev. A only.) Very occasionally, the LOCATE and GET instructions will garble a byte of memory. The most common place this happens is immediately after a READ statement, in which case it can garble one of the DATA statements in your program, but it can occur in a few other situations. The workaround is to ignore it until it happens to you. When it does happen, insert a dummy statement using the STR$ function (like DUMMY$=STR$(0)) immediately before the troublesome GET or LOCATE. This should clear things up. (Printing any number will also work because PRINT calls STR$ internally.)

Bug #4. (Rev. A only.) The interpreter will accept an INPUT or READ statement with no variable names after it! It should report an ERROR, of course. There's no workaround; you should simply be careful. Also, save any program before running it. If it locks up for no apparent reason, check to see if you've made this error.

Bug #5: (Rev. A only.) Unary operators don't quite work correctly, especially the NOT operator. PRINT NOT NOT 1 will produce an ERROR 10 (expression too complex) if you're lucky--or crash your computer if you're not. Unary minus also has a problem (unrelated to bug #2 above): PRINT --3 will print -3 rather than just 3. Workaround: Don't stack NOT's, -'s, or +'s without an intervening parenthesis. And in the case of NOT, use a parenthesis even if you're not stacking anything else. In other words, use -(-3) instead of --3, or NOT (A<B OR A<C) rather than trying to get away with NOT A<B OR A<C.

Interestingly, Atari's fix for this bug (in revs. B and C) is slightly different from the fix OSS uses in BASIC A+/XL/XE. Unary + and - were fixed the same way in all of them, so PRINT --3 will print 3, as it should. However, PRINT NOT NOT 1, which prints 1 on OSS's BASICs (as it should), generates a syntax error on Atari BASIC B or C! OSS's fix requires extensively restructuring BASI "operator precedence table," a task which was apparently beyond the programmers at the "old" Atari, so they simply made stacked NOTs illegal instead!

Bug #6: (Rev. B only.) The 16-byte "expando" bug.

Apparently, there is a bug in either the Atari OS or rev. A BASIC which occasionally causes them to attempt to write over each other's memory when memory is nearly full. In a lame attempt to work around this bug, some jerk at the "old" Atari decided that if they made BASIC think that a program was 16 bytes longer than it really was, they would avoid getting zapped by the OS. Unfortunately, they did this by adding 16 to each of the "page 0" pointers BASIC uses to keep track of where various parts of your program are. The net effect is that the 16 "extra" bytes are at the bottom of the program, where they are of absolutely no help in doing what they are designed to do (avoid the OS), but do manage to get themselves SAVEd each time you save the

program. The really bad thing is that they get LOADed back when you reload the program, but another 16 bytes are added at that time, so the effect is cumulative: each time you SAVE and then reLOAD a program, it gets 16 bytes larger! Eventually, you end up with a lot of wasted space in your programs.

The only workaround is to use LIST and ENTER rather than SAVE and LOAD. If you have rev. B, I strongly suggest you do this, because it will also help you recover when the infamous keyboard lockup bug strikes.

Bug #7: (All versions, including BASIC A+/XL/XE.) Believe it or not, STR$(A)=STR$(B) only compares the lengths of STR$(A) and STR$(B), not their values! This is an example of asking BASIC to be more intelligent than any 8K program is likely to be. You see, the STR$ function simply calls the "convert to ASCII" routine in the floating-point package, which always puts the resulting string in a buffer at address hex 580 in memory. So when the above expression is evaluated, the printed representation of A is stored at $580. Then STR$ is called again, which places the printed representation of B at $580, overlaying STR$(A). Now the two strings at $580 are compared. If they're the same length, of course they match!

To work correctly, STR$ would have to allocate memory and copy the result to it. But we can do that ourselves in this workaround:

```
A$=STR$(A):IF A$=STR$(B) THEN ...
```

Bug #8: (All versions, including BASIC A+/XL/XE.) This is my personal favorite. Amaze your friends; you can DIM A(32766,32766) without getting an "out of memory" error! In fact, this only reduces free memory by six bytes (the size of one array element)!

BASIC only does a two-byte multiply of your array bounds. If the product is over 64K, therefore, BASIC will only attempt to allocate the remainder (over the nearest multiple of 64K) and so may not notice the error. Then, when it calculates the address of a reference to the array, it can end up practically anywhere, causing a crash, a garbled program, or other strange symptoms. The workaround

is the same as for the INPUT/READ bug in Rev. A: Be careful, save your program before running, and if you can't figure out what's wrong, look for this error.

Bug #9: (All versions, including BASIC A+/XL/XE.) When PRINTing a quoted string ending in ctrl-R or ctrl-U, a carriage return is not output, as though a semicolon were at the end of the PRINT statement.

This bug makes no sense at all unless you understand the way BASIC represents your program in memory. Every line of your BASIC program except REMs and DATA statements is converted to a "tokenized" format when entered. In particular, each special symbol in BASIC (e.g., ", #, :, and even two-character symbols such as <=) is represented by a one-byte "token" rather than its ATASCII value. Constants are also tokenized: a numeric constant becomes a special "number" token followed by its six-byte floating-point value, and a string constant becomes a "string" token followed by a length byte and finally the contents of the string. But notice there is NO token for the CLOSING quotation mark of a string. So when BASIC gets to the end of a PRINT statement, and it looks to see if the previous token is a comma's or semicolon's token, so it can tell whether or not to print a carriage return, if the PRINT statement ends with a quoted string, BASIC ends up thinking the last character in the string is the previous token! And guess which characters correspond to the comma and semicolon tokens?

The easiest workaround is simply to print CHR$(18) or CHR$(21) instead of ctrl-R or -U. The problem can also occur when printing expressions ending in certain numeric literals. This is pretty rare, but if it does happen, simply assign the expression to a variable and print the variable instead; i.e., replace "PRINT X/.1212121212" with "ANS=X/.1212121212:PRINT ANS".

Bug #10: (All versions, including BASIC A+/XL/XE.) BASIC correctly diagnoses a reference to IOCB #8 thru #15 as an error. But #16 thru #23 work fine, having the same meaning as #0 thru #7 respectively.

This is an example of a "feature:" a bug with useful

B&C ComputerVisions

3283 Kifer Road
Santa Clara CA 95051
(408) 749-1003

SUPER SALE

RECONDITIONED ATARI MERCHANDISE

All merchandise has been tested and reconditioned (except where noted). The "A" price indicates like-new condition. The "B" price indicates product may have scratches and other superficial surface mars. 30 day warranty except where noted.

ATARI TRAK BALL
$8.95 B
$10.95 A
FAST ARCADE-LIKE ACTION

ATARI SPACE AGE JOYSTICK
$5.50 B
$6.50 A
LOOKS LIKE A GUN WITH A TRIGGER FOR GREAT SHOOT-EM-UP ARCADE ACTION. COMES IN ORIGINAL BOX.

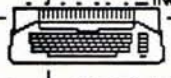
STANDARD ATARI JOYSTICK

$3.50 B
$4.50 A

2600 REMOTE CONTROL JOYSTICKS (2)
$12.95 B
$15.95 A
REQ. 2600 POWER SUPPLY TO USE ON 800/XL/XE - $5.00

HARD TO FIND TOUCH TABLET
$32.95 B
$39.95 A
IN STYROFOAM BOX

NUMERIC KEY PAD
$6.95 B
$7.95 A
USE WITH VISICALC, BOOKKEEPER & BASIC FOR FAST NUMERIC F[illegible]Y
INCLUDES HANDL[illegible]K



40 COLUMN 822 PRINTER
$34.95 B
$39.95 A
INCLUDES ROLL OF THERMAL PAPER

1030 MODEM WITH *EXPRESS*

$29.95 B
$34.95 A
IN STYROFOAM BOX
INCLUDES I/O CABLE & POWER SUP.

16K 400 COMPUTER
$24.95 B
$29.95 A
INCL. POWER SUPPLY AND TV SWITCH BOX
48K UPGRADE $25.00

800 48K COMPUTER
$69.95 B
$79.95 A
INCL. POWER SUPPY, TV SWITCH BOX, BASIC CARTRIDGE & BASIC MANUAL

850 INTERFACE
$79.95 B
$89.95 A
I/O CABLE & POWER SUPPLY
LIMITED SUPPLY

410 PROGRAM RECORDER
$9.95
IN ORIGINAL BOX - AS IS
NOT TESTED
NO WARRANTY

810 DISK DRIVE
$110.00 B
$120.00 A
COMPLETE WITH CASE, I/O, POWER SUP. & DOS 2

DISKETTES
AS LOW AS
20 CENTS
10 FOR $4.00
100 FOR $29.00
1000 FOR $200
UN-NOTCHED WITH DOS 3

THE BOOKKEEPER NEW IN BOX
$24.95
WITHOUT KEYPAD
$29.95
WITH KEYPAD

CARTRIDGES $5.00 EACH
MISSILE COMMAND
STAR RAIDERS
DONKEY KONG
DEFENDER
PAC-MAN
BASIC
SOME IN BOXES

SHIPPING INFORMATION

Prices do not include shipping and handling. Add $5.00 for small items Add $8.00 for disk drive. Calif. residents include 7% sales tax. Mastercard and Visa accepted if your telephone is listed in your local phone directory. Orders may be pre-paid with money order, cashier check, or personal check. Personal checks are held for two weeks before order is processed. C.O.D orders are shipped via UPS and must be paid with cash, cashier check or money order. International and APO orders must be pre-paid with cashier check or money order. $20.00 minimum on all orders. All sales are final - no refunds - prices are subject to change. Phone orders accepted TUESDAY THROUGH FRIDAY from 10:00 am to 6:00 pm PST.

We carry a complete line of ATARI products and have a large public domain library. Write or call for free catalogue. (408) 749-1003 TUE - FRI 10AM - 6 PM

s. effects. For instance, have you ever wanted to issue a simple INPUT statement without getting a "?" printed? With BASIC A+/etc. you can INPUT #0, but if you use Atari BASIC, you were stuck. Oh sure, you could always OPEN an IOCB to the E: device, but that cleared the screen. So you probably ended up OPENing an IOCB to the K: device, then using GET and PRINT to assemble the input one character at a time.

Now you don't have to. To INPUT from the screen without a "?" printing, simply INPUT #16. Since #16 is the same as #0, and since IOCB #0 is always open to the screen, this will work.

That's all the bugs I know of. There could be others, of course. Now that you know all the pitfalls, you can decide whether it's worth it to upgrade to rev. C or one of the OSS BASICs, and in the meantime, you can program more confidently.

See you next month.

ACCOKC is a non-profit, tax empt and independent computer club dedicated to educating our members in the use of the ATARI computer. We have been allowed to use the word 'ATARI' in the identification of our computer club by ATARI CORP., but are in no way affiliated. Other club newsletters may reprint from this publication as long as authors and our club are given due recognition in the article. Articles may be submitted via modem by calling 751-8955. Use Xmodem protocol and please include your full name and telephone number.

For help or information call:

President:	Bob Bewley	751-8955
Sec/Treas:	Becky Pound	942-5100
Librarian:	Tom Holden	789-4379
Editor :	Unknown Word Processor !!	

ATARI COMPUTER CLUB
of OKLAHOMA CITY
P. O. Box 32672
OKLAHOMA CITY, OK.
73123